Architettura degli elaboratori A.A. 2007-2008

Quinto appello - 9 settembre 2008

Domanda 1

Un programma scritto in LC si compone di due processi F e C. Il processo F riceve da C richieste di calcolo di una funzione *f(X,Y)* con X e Y di tipo vettori di interi da 32 bit da 1K posizioni ciascuno. Le richieste avvengono utilizzando un canale asincrono con grado di asincronia 1 e le relative risposte vengono restituite al processo C utilizzando ancora un canale asincrono con lo stesso grado di asincronia. Si dettagli il codice LC dei due processi e si indichi l'effetto della loro esecuzione sulla schedulazione, assumendo che l'invio del messaggio di richiesta da parte di C provochi (o alternativamente *non* provochi) un fault di pagina nell'accesso alle strutture dati che rappresentano i vettori X e Y.

Facoltativo: si consideri l'effetto di qualunque fault di pagina generato nell'esecuzione del codice LC.

Domanda 2

Si consideri l'ipotesi che il calcolo di un polinomio di secondo grado ax^2+bx+c rappresenti un calcolo molto frequente in una serie di applicazioni da eseguire su un processore che interpreta l'assembler D-RISC. Si estenda quindi l'assembler D-RISC con un'istruzione

POL2 R_p,R_x,R_r

che calcola

$R_r = R_p * R_x * R_x + R_{p+1} * R_x + R_{p+2}$

(cioè il polinomio di secondo grado i cui coefficienti sono rappresentati nei 3 registri a partire da R_p) dettagliando il codice per le microistruzioni che la interpretano, fornendone il tempo medio di elaborazione e discutendo le implicazioni sulla parte operativa. Per la soluzioni proposta si valutino le differenze, in termini di risorse richieste e di tempo medio di elaborazione rispetto all'esecuzione dello stesso calcolo utilizzando D-RISC puro. Si assuma la disponibilità di ALU in grado di effettuare moltiplicazioni fra parole in $10t_p$.

Domanda 3

1) Si fornisca lo schema di una rete sequenziale secondo il modello di Mealy che implementa il calcolo del checksum di uno stream di parole da 32 bit. Il checksum di un insieme di parole $w_1, \ldots, w_k$ è ottenuto sommando tutte le parole senza tenere in considerazione eventuali overflow.

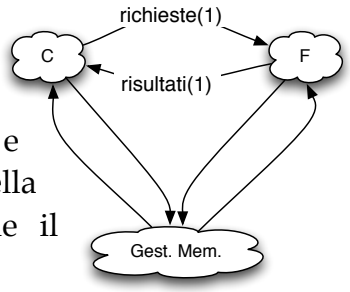
2) Si discuta l'impatto della ampiezza del blocco di una memoria cache associativa su insiemi sul numero di fault che si verificano durante l'esecuzione di un programma.

ATTENZIONE: Riportare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso.
I risultati e il calendario degli orali saranno pubblicati sulla pagina WEB del corso appena disponibili.

Bozza di soluzione

Domanda 1

La strutturazione a processi è quella riportata in Figura.



Assumiamo che la capacità dei canali sia tale da poter permettere l'invio delle 2K parole relative ai parametri della funzione f e all'invio della corrispondente risposta contente il risultato della chiamata di funzione. Il codice dei due processi sarà dunque il seguente:

```
C::   channel in risultati(1), channel out richieste;
      ...
      // si riporta solo la parte in cui il client
      // interagisce con il server di calcolo di f
      send(richieste,<X,Y>);
      receive(risultati,<risultato,esito>);
      if(esito != "OK")
           trattamento_errore();
      ...

F::   channel in richieste(1), channel out risultati;
      {
      // è un server, quindi ciclo infinito ...
      while(true) {
           receive(richieste,<X,Y>);  // ricevi richiesta
           <ris,esito> = f(X,Y);      // effettua calcolo
           send(risultati,<ris,esito>); // invia risposta
           }
      }
```

Dal punto di vista della schedulazione, assumendo che l'accesso ai vettori X e Y *non* provochi un fault di pagina su C (per esempio perché i vettori erano stati precedentemente "calcolati" in qualche modo e non erano ancora stati eliminati dalla cache al momento della richiesta di calcolo della funzione):

i. C diventa il primo processo della lista dei processi pronti

ii. C va in esecuzione ed esegue la send. Il processo F sarà in attesa sulla receive, quindi nello stato di attesa. L'esecuzione della send provoca la copia del messaggio (<X,Y>) nella variabile targa di F e la sveglia processo per F (F finisce in nella lista dei processi pronti). C esegue la receive e si blocca finendo nello stato di attesa. Il processore passa al prossimo processo della lista dei pronti

iii. F arriva nella prima posizione della lista dei processi pronti ed ottiene l'uso del processore non appena il processo corrente viene deschedulato. Esegue il calcolo di F e successivamente la send verso C. C è in attesa. Si può effettuare la copia "alla pari" del messaggio nella variabile targa. Si effettua una sveglia processo per C. C finisce nuovamente nella lista dei processi pronti.

ATTENZIONE: Riportare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso.
I risultati e il calendario degli orali saranno pubblicati sulla pagina WEB del corso appena disponibili.

Se invece si assume che l'accesso ad X (per esempio) provochi un fault di pagina, il passo ii. si espande in:

ii. C esegue la send. Nel codice della send, al momento della copia alla pari del msg nella variabile targa si ha un fault di pagina. L'eccezione viene rediretta sul gestore della memoria con una send di un messaggio opportuno. Il processo C viene deschedulato in attesa di una risposta dal gestore della memoria. Dopo che la pagina è stata caricata e la tabella di rilocazione per C aggiornata di conseguenza, il gestore di memoria effettua una comunicazione verso C che lo sblocca, rendendolo di nuovo pronto per l'esecuzione. L'esecuzione della send da C su F riprende con il riavvio della copia del messaggio (<X,Y>) nella variabile targa di F (questa volta e la sveglia processo per F (F finisce in nella lista dei processi pronti). C esegue la receive e si blocca finendo nello stato di attesa. Il processore passa al prossimo processo della lista dei pronti.

Facoltativo: Fault di pagina nell'esecuzione di C possono verificarsi anche in corrispondenza dell'accesso alla struttura dati canale richieste, o per l'accesso al codice LC del processo stesso. In entrambi i casi la schedulazione comporta il passaggio nello stato di pronto per il processo Gestore della memoria e nello stato di attesa per il processo C, esattamente come nel caso del fault dovuto all'accesso alle strutture dati.

Domanda 2

Vediamo innanzitutto come implementare l'istruzione nell'interprete firmware. Le due istruzioni di decodifica sono quelle della dispensa. Senza tener conto di limitazioni sul numero delle ALU disponibili nella parte controllo, potremmo interpretare la POL2 in un'unica microistruzione, come segue:

```
pol2_0:   (INT=0) RG[IR.operando1] * RG[IR.operando2] * RG[R.operando2] +
          RG[IR.operando1+1]*RG[IR.operando2] +
          RG[IR.operando1+2] → RG[IR.operando3], IC+1 → IC, ch0
          (INT=1) RG[IR.operando1] * RG[IR.operando2] * RG[R.operando2] +
          RG[IR.operando1+1]*RG[IR.operando2] +
          RG[IR.operando1+2] → RG[IR.operando3], IC+1 → IC, tratt_int
```

Questo richiede complessivamente 7 ALU (comprensive delle due ALU necessarie per calcolare IR.operando1+x, con x=1,2). Se più realisticamente consideriamo la disponibilità di tre sole ALU nella parte operativa (due per operazioni "lunghe" e una per gli offset nell'indirizzamento del file dei registri) le microistruzioni che implementano la POL2 diventano 3:

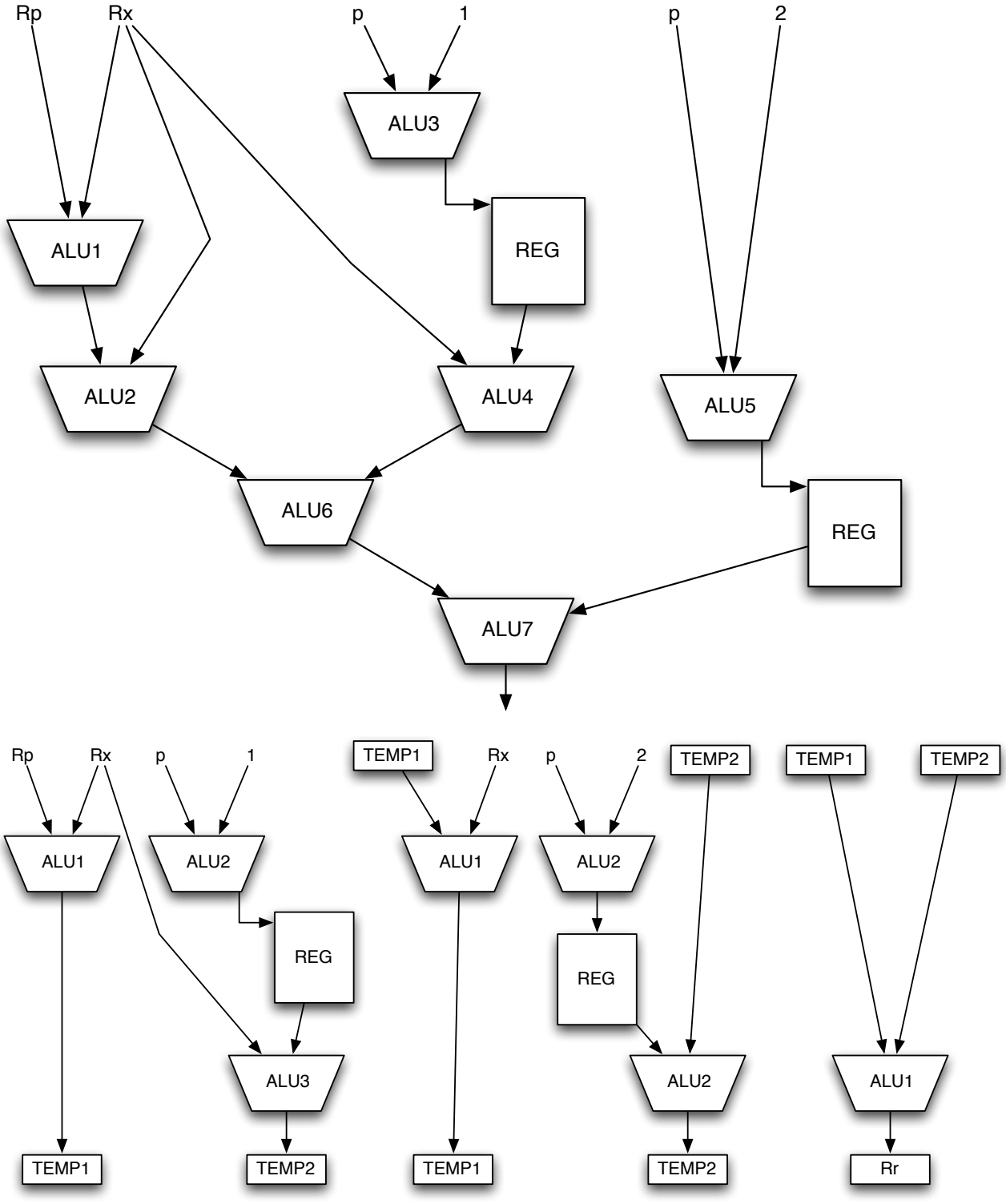
```
pol0:   RG[IR.operando1] * RG[IR.operando2] → TEMP1,
        RG[IR.operando1+1]*RG[IR.operando2] → TEMP2, 1

pol1:   TEMP1 * RG[R.operando2] → TEMP1, TEMP2 + RG[IR.operando1+2] → TEMP2

pol2:   (INT=0) TEMP1 + TEMP2 → RG[IR.operando3],IC+1→IC, ch0
        (INT=1) TEMP1 + TEMP2 → RG[IR.operando3],IC+1→IC, tratt_int
```

ATTENZIONE: Riportare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso.
I risultati e il calendario degli orali saranno pubblicati sulla pagina WEB del corso appena disponibili.

Le due implementazioni corrispondono alla parte alta e bassa della Figura che segue, rispettivamente (sulle ALU mancano tutti i commutatori e i segnali di controllo. I registri per i quali non occorre "calcolare" l'indirizzo, sono riportati senza logica di accesso).



ATTENZIONE: Riportare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso.
I risultati e il calendario degli orali saranno pubblicati sulla pagina WEB del corso appena disponibili.

Nel primo caso, il cammino più lungo nella PO passa attraverso 4 ALU (una per il calcolo di IC.operando+x, e le altre 3 per i tre livelli dell'albero dell'espressione), per un totale di almeno $40t_p$. Questo tempo implica che il ciclo di clock del processore deve essere dimensionato in modo da non essere inferiore a

$$T\omega_{PO} + \max\{T\omega_{PC}+T\sigma_{PO}, T\sigma_{PC}]+\delta$$

dove questa volta $T\sigma_{PO} \geq 40t_p$. Nel secondo caso, invece, si utilizzano al massimo due ALU in cascata, una per il calcolo di IR.operando+x e l'altra per il calcolo del risultato parziale. Dunque la formula sopra vale per $T\sigma_{PO} \geq 20t_p$. Ovviamente dovranno essere presi ragionevolmente in considerazione i commutatori all'ingresso delle ALU, per un contributo pari a $4*2t_p$ nel primo caso e $2*2t_p$ nel secondo caso. L'implementazione della PO non varierà ragionevolmente nei due casi, anche se nel primo caso avremo un maggior numero di variabili di controllo per la gestione di tutte le ALU e dei relativi commutatori. Considerando il primo e il secondo caso, comunque, l'istruzione verrà eseguita in $2\tau+t_a$ (tempo per la chiamata istruzione) + 1 (o 3) τ per la sua esecuzione effettiva.

Nel caso avessimo utilizzato D-RISC puro, avremmo comunque dovuto tener conto dell'assunzione che si potessero utilizzare ALU che implementano la moltiplicazione fra parole in $10t_p$ (per uniformità), e quindi avremmo dovuto assumere l'esistenza di una

MUL R_a, R_b, R_c ; Ra * Rb → Rc

Di conseguenza la POL2 si sarebbe potuta implementare con una sequenza

```
MUL Rp,Rx,Rterm1
MUL Rx,Rterm1,Rterm1
MUL Rx,Rp+1,Rterm2
ADD Rterm2,Rterm1,Rtermleft
ADD Rtermleft,Rp+2,Rr
```

per la quale si sarebbero spesi complessivamente

$5*(2\tau+t_a + \tau)$

Questo valore non è ragionevolmente confrontabile con il valore della prima soluzione fw, dal momento che tale soluzione innalza sostanzialmente la durata del ciclo di clock, ma può essere ragionevolmente confrontata con la seconda soluzione (che utilizza un numero comparabile di ALU in cascata). In questo caso dunque avremmo $15\tau+5t_a$ contro i $5\tau+t_a$ della implementazione a firmware. L'implementazione a firmware dell'istruzione POL2 porterebbe quindi ad un guadagno di prestazioni pari a $10\tau+4t_a$ per ogni sua esecuzione.

Quanto detto finora non tiene in considerazione la possibilità di avere un overflow nell'esecuzione delle operazioni di moltiplicazione. Se si verificasse un overflow dovremmo invocare il trattamento eccezioni, esattamente come nel caso in cui si verifichi un esito non nullo in una lettura in memoria. Consideriamo solo quanto sarebbe necessario fare nel secondo caso (uso di due sole ALU in cascata) per

trattare correttamente questa condizione. Il microcodice dovrebbe essere modificato come segue:

pol0: RG[IR.operando1] * RG[IR.operando2] → TEMP1,
RG[IR.operando1+1]*RG[IR.operando2] → TEMP2, pol1

pol1: (OW1,OW2=00) TEMP1 * RG[R.operando2] → TEMP1,
TEMP2 + RG[IR.operando1+2] → TEMP2, pol2
(=--) tratt_eccezione

pol2: (INT,OW1,OW2=000) TEMP1 + TEMP2 → RG[IR.operando3],IC+1→IC, ch0
(=100) TEMP1 + TEMP2 → RG[IR.operando3],IC+1→IC, tratt_int
(=---) tratt_eccezione

In questo caso assumiamo che OWi sia il flag di overflow relativo alla ALUi utilizzata per l'implementazione della operazione di cui si vuole controllare l'esito. Da notare che in questo codice non si controlla l'eccezione dovuta alla somma delle due parti del polinomio calcolata nella microistruzione pol2. Se si volesse tener conto anche dell'overflow nell'ultima operazione di somma, potremmo introdurre un'ulteriore microistruzione, nella quale testare l'overflow:

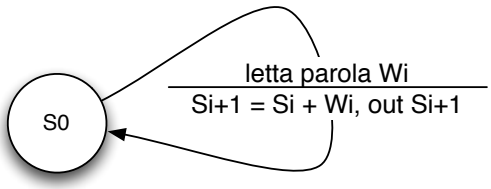
pol2: (OW1,OW2=00) TEMP1 + TEMP2 → RG[IR.operando3], pol3
(=---) tratt_eccezione

pol3: (INT,OW1=00) IC + 1 → IC, ch0
(=01) tratt_eccezione
(=1-) IC + 1 → IC, tratt_int

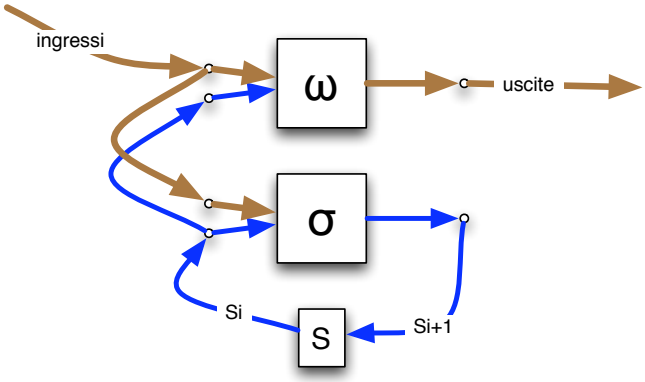
con ovvie conseguenze sul tempo necessario per completare l'operazione.

Domanda 3

Punto 1. L'automa è il seguente:



Vi è un'unica variabile di stato, S che contiene il valore del checksum come calcolato fino a quel momento. La rete sequenziale di Mealy nella sua forma generale ha lo schema:



ATTENZIONE: Riportare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso.
I risultati e il calendario degli orali saranno pubblicati sulla pagina WEB del corso appena disponibili.

In questo caso sia omega che sigma calcolano Si+Wi (valore dello stato corrente sommato (modulo 32bit) all'ingresso.

Punto 2. (vedi dispensa)
L'aumento dell'ampiezza del blocco di una cache associativa su insiemi determina una riduzione del numero di fault dovuti agli accessi a parole consecutive nelle memoria logica del processo. Tuttavia, a parità di capacità totale della cache, l'aumento dell'ampiezza del blocco determina una diminuzione del numero di blocchi stessi e quindi in sostanza un aumento dei fault dovuti ad accessi a blocchi distinti nel working set del processo.